VMC^{++}quick start guide

Iwan Kawrakow

June 4, 2010

1 Installation

1. Unzip and untar the distribution file:

   ```
   % tar -xjvf vmc.tar.bz2
   ```

2. Define the environment variable `vmc_home` to point to the VMC^{++} installation directory: put

   ```
   export vmc_home=path_to_vmc   (if using bash)
   setenv vmc_home path_to_vmc   (if using tcsh)
   ```

   into your `.bashrc` or `.cshrc` file.

3. Define the environment variable `vmc_dir` to point to the directory where you will have your input and output files for VMC^{++}calculations. This may be the folder `runs` in the installation directory, which has several example input files, or any other folder.

4. The VMC^{++}executable uses a shared library (`libvmcpp.so` in `$vmc_home/bin`). In order this library to be found, you have to

   - Move `libvmcpp.so` to a standard library directory, *e.g.* `/usr/local/lib` (if you have root privileges), or
   - Include `$vmc_home/bin` in the library search path, *e.g.* put

     ```
     export LD_LIBRARY_PATH=$vmc_home/bin:$LD_LIBRARY_PATH
     ```

into your `.bashrc` file, or

- Always run VMC^{++}from the `$vmc_home/bin` directory

5. You may include the `$vmc_home/bin` directory into your search path for convenience.

2 Running VMC^{++}calculations

To run a VMC^{++}calculation, use

```
% vmcpp.exe -i input_file [-o output_file] [-j job_number]
```

or

```
% vmcpp.exe -r job_control_file [-j job_number]
```

where `input_file.vmc` is a file in `$vmc_dir` that specifies geometry, source, etc. for a VMC^{++}simulation and where `some_file.run` is a file in `$vmc_dir` that contains calculation requests for VMC^{++}(note that the extension `.run` or `.vmc` should not be given on the command line). Command line parameters given in square brackets are optional. With the `-j` command line option one can change the random number sequence (this is intended for parallel runs). The first command version can be used to perform a single VMC^{++}simulation, whereas the second is useful for performing a series of VMC^{++}simulations as specified in the job control file. The format of this file is as follows:

- Lines beginning with a pound sign (#) are ignored and can be used for comments.
- Lines of the form

  ```
  calculate:  some_input_file
  ```

  request a fresh calculation from VMC^{++}with the parameters, geometry and source specified in the input file `some_input_file.vmc` (see next section for the format of input files).
- Lines of the form

  ```
  restart:  some_number
  ```

request a continuation of a previous calculation. The floating point or integer number `some_number` is interpreted as follows: If `0` $<$ `some_number` $<$ `100`, the calculation will be run until the average relative uncertainty in the region $D > D_{\rm max}/2$ is better than `some_number`%. Otherwise additional `some_number` particle histories will be run. Obviously a restart request can not be first request in a run control file.

For instance, the sequence

```
calculate:  test1
restart:  100000
restart:  0.5
calculate:  junk/test2
restart:  1.5
```

will make VMC^{++}run a calculation as specified in the input file `$vmc_dir/test1.vmc` and output the results, then run additional 100000 particles using the same transport parameter, geometry and source and output the combined results from the first and second run, then continue the calculation until the statistical uncertainty is less than 0.5% and output the combined results from all 3 runs (if the uncertainty is already less than 0.5% after the second run, no additional particles will be simulated). Next, a fresh calculation as specified in the input file `$vmc_dir/junk/test2.vmc` will be started, the result output, and another calculation with the `$vmc_dir/junk/test2.vmc` parameter will be run until the uncertainty is better than 1.5%.

3 Input files

All VMC^{++}input files must be in the `$vmc_dir` directory (but you may have subdirectories in `$vmc_dir` as shown above). An input file specifies transport parameter, geometry, source and calculation options. In order to make the understanding of the syntax of an input file easier, several sample input files have been included in the distribution:

- `iccrp18mv.vmc`: input file for the ICCR 2000 18 MV photon beam accuracy benchmark
- `iccre20mev.vmc`: input file for the ICCR 2000 20 MeV electron beam accuracy benchmark
- `head_e12_a06.vmc`: input file for a calculation using a density matrix for geometry definition (see below) and a 12 MeV electron beam
- `head_p06.vmc`: same as above but now using 2 6 MV photon beams.

3.1 Input file syntax

Sought input is identified with "code words". The input following a code word is then extracted and passed to the object looking for the input. Every VMC^{++}object looks for relevant input only within a section of the input file that is separated by a start delimeter and a stop delimeter. For instance, the delimeter of the geometry factory is `geometry` and so, the geometry factory only looks for relevant information only within the part of the input file that is enclosed by

```
:start geometry:
:stop geometry:
```

The following general syntax rules apply for any VMC^{++}input

1. The order in which code words and sections (a part of the input file enclosed by a pair of `start` and `stop` delimeter) appear in the input file is irrelevant.
2. Code words must be followed by `'='` or `':'`, blanks between the actual code word and the `=` or `:` sign are permitted, *i.e.*
   ```
   my code word =
   my code word=
   my code word     :
   ```
   are all the same.
3. The parser is not case sensitive, *i.e.*
   ```
   my code word =
   My CoDe WOrd =
   MY CODE WORD =
   ```
   are all the same.
4. Input following `'#'` (Unix comment) or `'!'` (Fortran comment) is ignored.
5. Input can be integers, floating numbers or strings
6. If multiple input per code word is to be parsed, the input tokens must be separated by commas or blanks *i.e.*
   ```
   my code word = 1, 2, 3
   ```
   or
   ```
   my code word = 1  2  3
   ```
   or
   ```
   my code word = aluminum, water  bone
   ```

7. If input is to be continued on the next line, the line must end with a comma or backslash (Unix line continuation), *e.g.*
```
 my code word =  aluminum,
                 water,
                 bone
```

3.2 Geometry definition

The geometry definition must be between delimeters `geometry`. In principle, VMC^{++}can run simulations that involve several geometries at once, but this goes beyond the scope of this quick start guide. Because of this, the input for each geometry must be separated on its own right by a delimeter that depends on the geometry type being constructed. For XYZ geometries the delimeter is `xyz geometry`. There are 2 methods to define a XYZ geometry:

- Via direct definition of planes and media in the input file, see `iccre20mev.vmc` and `iccrp18mv.vmc` for examples.
- Via a density matrix file. The format of the density matrix file can be seen from the little C++ program `$vmc_home/phantoms/read_density_matrix.cpp`. This folder also contains a sample density matrix of a human head and neck. It is up to you to convert a CT image into the density matrix file format needed by VMC^{++}and it is up to you to decide upon the Hounsfield number to mass density conversion rule (which depends on the CT energy). The mass density to material conversion rule is defined in the data file `$vmc_home/data/ct_ramp.data`. This file is simply a list of media names (see next section to learn about defining media for use with VMC^{++}), minimum and maximum density for which to use this medium and its default density. You can edit `ct_ramp.data` to fit your needs. Examples of using a density matrix file for defining the geometry can be found in `head_e12_a06.vmc` and `head_p06.vmc`.

3.3 Source definition

Source input must be between the delimeter `general source`. A general source can have an arbitrary number of sources. The general source object searches the text between the delimeters for occurrences of

```
:start some source:
some text
:stop some source:
```

removes this from the text and attempts to load a DLL with the name `libsome_source.so` in the `$vmc_home/sourcelib` directory. If successful, it passes the text between the `some source` delimeters to the source creation function of the DLL. The process is continued until no text is left. In this way, one can have an arbitrary number of sources in a simulation and also one can implement his/her own source plugins for VMC^{++}(if you are interested in doing so, I can send you the relevant header files and specs). The distribution contains right now only DLL's for a point source (`libsimple_source.so`) and for a BEAM phase space file source (`libphsp_source.so`). The code words for the definition of a point source (called simple source and implemented in `libsimple_source.so`) should be self-explanatory together with the example input files. It is important to note that the monitor units for each source must be specified in the `general source` section, even if just one source is used. The MU input can be used to renormalize the output of a VMC^{++}simulation. The result output is dose per incident "fluence" times incident "fluence" per monitor unit times monitor units specified in the input file. Each source has its own definition of "fluence" and "fluence" per monitor unit. In the case of the point source, "fluence" is real fluence (*i.e.* particles per area) and "fluence" per MU is defined as 1. In the case of a phase space file source, it will be more sensible to define "fluence" as the number of electrons incident on the target and one may determine MU per "fluence" from calculated and measured dose per MU at a point. The general source can perform transformations of the particle positions and directions as coming from the various sub-sources. Such transformations are specified by including code words of the form

```
translation some_source = 0 5 -2
isocenter some_source = 7 15 6
angles some_source = 30 60 90
```

into the general source section, where `some_source` is the name of the source to be transformed (see `.vmc` files in the `runs` subdirectory for examples). If we denote with $\vec{x}$ and $\vec{u}$ the position and direction of the particles coming from `some_source`, with $\vec{x}_0$ the translation specified by the `translation some_source` keyword, with $\vec{x}_{\rm iso}$ the isocenter position specified by the `isocenter some_source` keyword and with α (first input), β (second input) and γ (last input) the angles specified by the `angles some_source` keyword, then the position $\vec{x}'$ and direction $\vec{u}'$ after the transformations will be

$$\begin{aligned} \vec{x}' &= R\left(\vec{x} + \vec{x}_0 - \vec{x}_{\rm iso}\right) + \vec{x}_{\rm iso} \\ \vec{u}' &= R\vec{u} \\ R &= R_x(\alpha)R_y(\beta)R_z(\gamma) \end{aligned} \tag{1}$$

with R_x, R_y and R_z rotation matrices around the x-, y- and z-axis. Confused? Well, I'm confused all the time too when it comes to source transformations. So, if you have a better suggestion let me know and I will implement it.

3.4 Transport parameter

Complete discussion of the various VMC^{++}transport parameter and how they influence accuracy/speed goes beyond the scope of the quick start guide. It is therefore best to use the automatic parameter selection (see the example input files).

3.5 Variance reduction

Same comments as above. Best is to use the input from the example files and to note that this is slightly different for electron and photon beams.

3.6 Number of particles, etc.

The number of particles to be run can be specified in the section delimited by `MC control`. It is also possible to request a calculation that runs until the specified statistical uncertainty is obtained (see `head_p06.vmc`.

3.7 Scoring options

The section delimited by `dose options` within the `scoring options` section defines in which geometry to score the dose and whether to calculate real dose or "dose to water".

3.8 Output

The amount and type of output is determined in the input section `output options geometry_name`, where `geometry_name` is the name of the geometry (because in principle there may be several geometries in a simulation, the output and scoring options for each geometry must be defined separately using the name of the geometry in the delimeter). Output of the entire dose matrix with uncertainty will be made if `dump=1`. If `dump=1`, the dose and dose uncertainty arrays will be dumped to a binary file called `input_file.dos`, where `input_file.vmc` is the name of the input file (if running a single calculation with the `-i` option, the output file can be set to `output_file.dos` with the `-o` command line option). The format of the binary file is:

- number of regions `nreg` (32 bit signed integer)
- `nreg` single precision floating numbers for the dose

- `nreg` single precision floating numbers for the dose uncertainty

A region with indeces `(ix,iy,iz)` has the address `ix + iy*nx + iz*nx*ny` in the 3d dose matrix.

One can also request the output of an arbitrary number of dose profiles by including input of the type

```
scan xo:  x1, y1, z1,   x2, y2, z2, ...
scan uo:  u1, v1, w1,   u2, v2, w2, ...
```

in the output options section. In the above, `x1,y1,z1,u1,v1,w1`, etc, define lines that pass through the points `x1,y1,z1`, etc., and have the direction cosines `u1,v1,w1`, etc. This somewhat cumbersome procedure to request a dose profile is motivated by the desire for generality (it can be used to output 1d plots of the dose variation in any geometry, not just XYZ, and it can be also used to plot the dose along arbitrary lines, not just along the x-, y- or z-direction). For each of the `xo,uo` pairs VMC^{++}will output a separate file called `input_file.profX`, where `X` is 0,1,etc. In these files the "position" is distance from the point `x1,y1,z1`, etc.

3.9 Smoothing

Smoothing can be turned on by including

```
:start smoothing options:
use smoothing = 1
:stop smoothing options:
```

as a separate section in the input file. One can control the maximum smoothing windows size in 3D using the code word `maximum 3D window`, the maximum smoothing windows size in 1D using the code word `maximum 1D window` and the maximum acceptable χ^2 using `maximum chi2` (note that the window sizes are actually half-window sizes). The smoothing algorithm included in this distribution is slightly different from the algorithm published in PMB. It applies the smoothing procedure iteratively several times and does not use batches. I have found that this gives similar or sometimes better performance than batching. The maximum number of iterations is controlled via `iterations`. If smoothing is turned on, then

- A binary 3D dose distribution will be output into a `.smoothed_dos` file. Note that this file is in the same format as the `.dos` file and therefore contains uncertainties, but these uncertainties should not be taken seriously (they are wrong).

- A forth column containing the smoothed dose will be output in each of the `.profX` files.

4 Media

VMC^{++}can perform simulations in any element between 1 and 100 and in any compound or mixture that is defined in the file `$vmc_home/data/media.data`. All cross section data necessary at run time is generated on-the-fly, once the media composition of the geometry is known. Elements can be identified either as integers corresponding to their atomic number, using their chemical symbols or using their names as defined in `$vmc_home/data/elements.data` (case sensitive). For instance, 13, Al or `ALUMINUM` are all valid options for using aluminum as a medium in the geometry. Other media are identified either with their id's or their names (see below). The format of the media definition file is as follows: first line contains medium id (an integer greater than 100 and greater than all previously used id's), medium name (case sensitive), mass density in g/cm^3, mean ionization energy in eV, number of elements (n) and 0 [1]. This is followed by n lines containing atomic number, atomic weight and fraction by weight for the n elements. As distributed, the `media.data` file contains several materials with compositions taken from ICRU-46 and various carbons that I have used for dosimetry related calculations. You can add any other material that you may need to `media.data`.

[1]This last number actually identifies whether the material is a conductor (1) or an insulator (0), which makes a difference in the calculation of the density effect. However, VMC^{++}does not support mixtures to be conductors at this point and therefore this number should be set to zero for all mixtures.